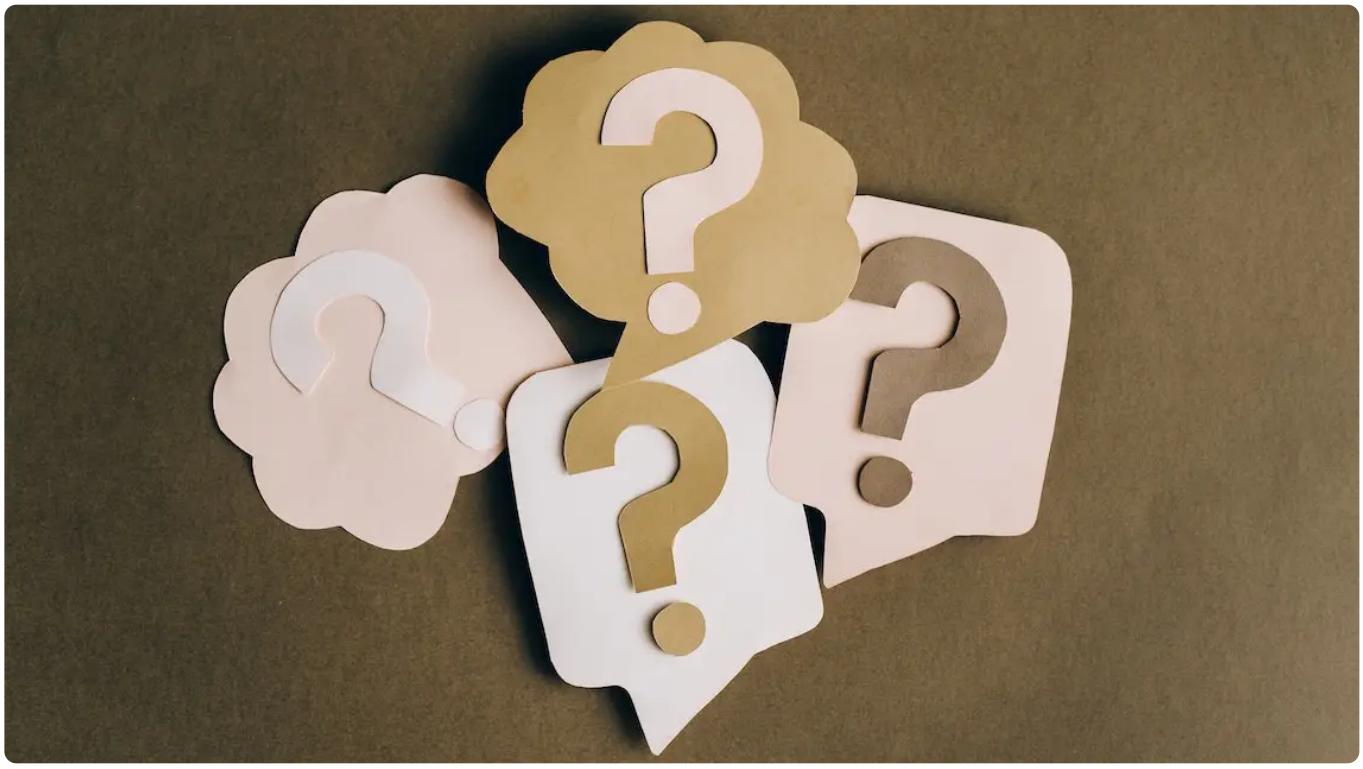
# 01 | IAM系统概述：我们要实现什么样的 Go 项目？

2021-05-26 孔令飞

《Go 语言项目开发实战》 课程介绍 >



**讲述：孔令飞**
时长 16:53 大小 15.47M

你好，我是孔令飞。从今天开始我们进入课前准备阶段，我会用 3 讲的时间给你讲清楚，我们要实现的实战项目 IAM 应用长啥样、它能干什么，以及怎么把它部署到 Linux 服务器上。先和我一起扫除基础的障碍，你就能够更轻松地学习后面的课程了。

今天这一讲，我先来说说为什么选择 IAM 应用，它能实现什么功能，以及它的架构和使用流程。

## 项目背景：为什么选择 IAM 系统作为实战项目？

我们在做 Go 项目开发时，绕不开的一个话题是安全，如何保证 Go 应用的安全，是每个开发者都要解决的问题。虽然 Go 应用的安全包含很多方面，但大体可分为如下 2 类：

**服务自身的安全**：为了保证服务的安全，需要禁止非法用户访问服务。这可以通过服务器层面和软件层面来解决。服务器层面可以通过物理隔离、网络隔离、防火墙等技术从底层保证服务的安全性，软件层面可以通过 HTTPS、用户认证等手段来加强服务的安全性。服务器层面一般由运维团队来保障，软件层面则需要开发者来保障。

**服务资源的安全**：服务内有很多资源，为了避免非法访问，开发者要避免 UserA 访问到 UserB 的资源，也即需要对资源进行授权。通常，我们可以通过资源授权系统来对资源进行授权。

**总的来说，为了保障 Go 应用的安全，我们需要对访问进行认证，对资源进行授权**。那么，我们要如何实现访问认证和资源授权呢？

认证功能不复杂，我们可以通过 JWT （JSON Web Token）认证来实现。授权功能比较复杂，授权功能的复杂性使得它可以囊括很多 Go 开发技能点。因此，在这个专栏中，我将认证和授权的功能实现升级为 IAM 系统，通过讲解它的构建过程，给你讲清楚 Go 项目开发的全部流程。

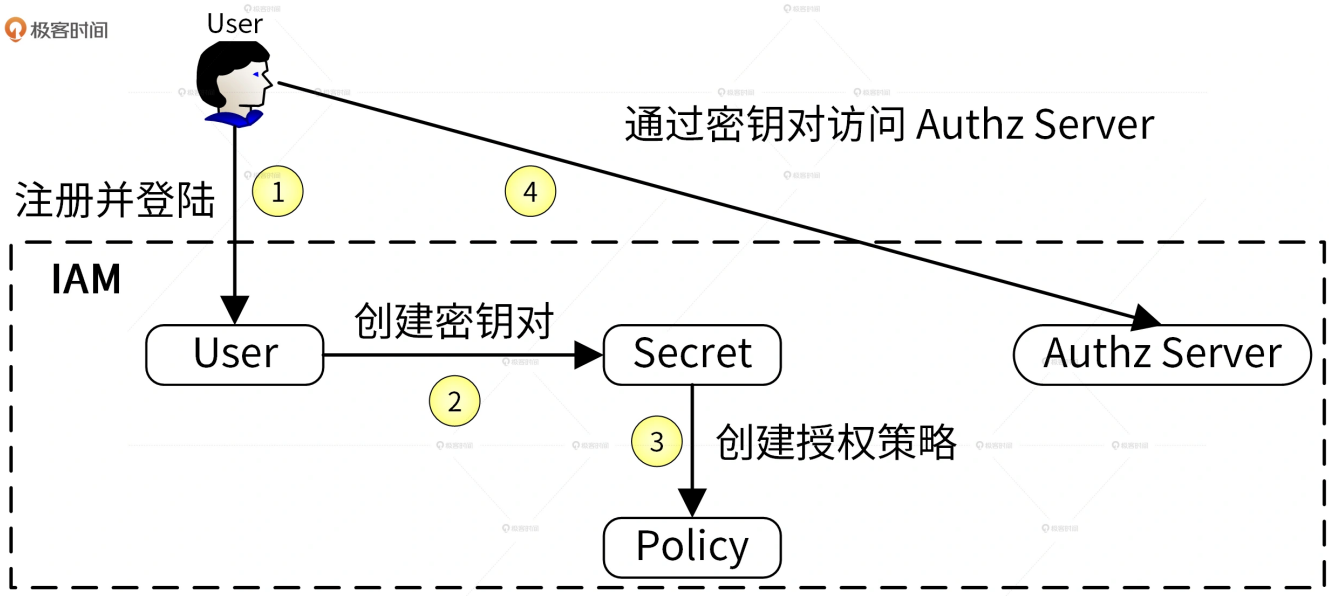
## IAM 系统是什么？

IAM（Identity and Access Management，身份识别与访问管理）系统是用 Go 语言编写的一个 Web 服务，用于给第三方用户提供访问控制服务。

IAM 系统可以帮用户解决的问题是：**在特定的条件下，谁能够 / 不能够对哪些资源做哪些操作**（**Who** is **able** to do **what** on **something** given some **context**），也即完成资源授权功能。

> 提示：以后我们在提到 IAM 系统或者 IAM 时都是指代 IAM 应用。

那么，IAM 系统是如何进行资源授权的呢？下面，我们通过 IAM 系统的资源授权的流程，来看下它是如何工作的，整个过程可以分为 4 步。




IAM 系统的功能示意图

1. 用户需要提供昵称、密码、邮箱、电话等信息注册并登录到 IAM 系统，这里是以用户名和密码作为唯一的身份标识来访问 IAM 系统，并且完成认证。
2. 因为访问 IAM 的资源授权接口是通过密钥（secretID/secretKey）的方式进行认证的，所以用户需要在 IAM 中创建属于自己的密钥资源。
3. 因为 IAM 通过授权策略完成授权，所以用户需要在 IAM 中创建授权策略。
4. 请求 IAM 提供的授权接口，IAM 会根据用户的请求内容和授权策略来决定一个授权请求是否被允许。

我们可以看到，在上面的流程中，IAM 使用到了 3 种系统资源：用户（User）、密钥（Secret）和策略（Policy），它们映射到程序设计中就是 3 种 RESTful 资源：

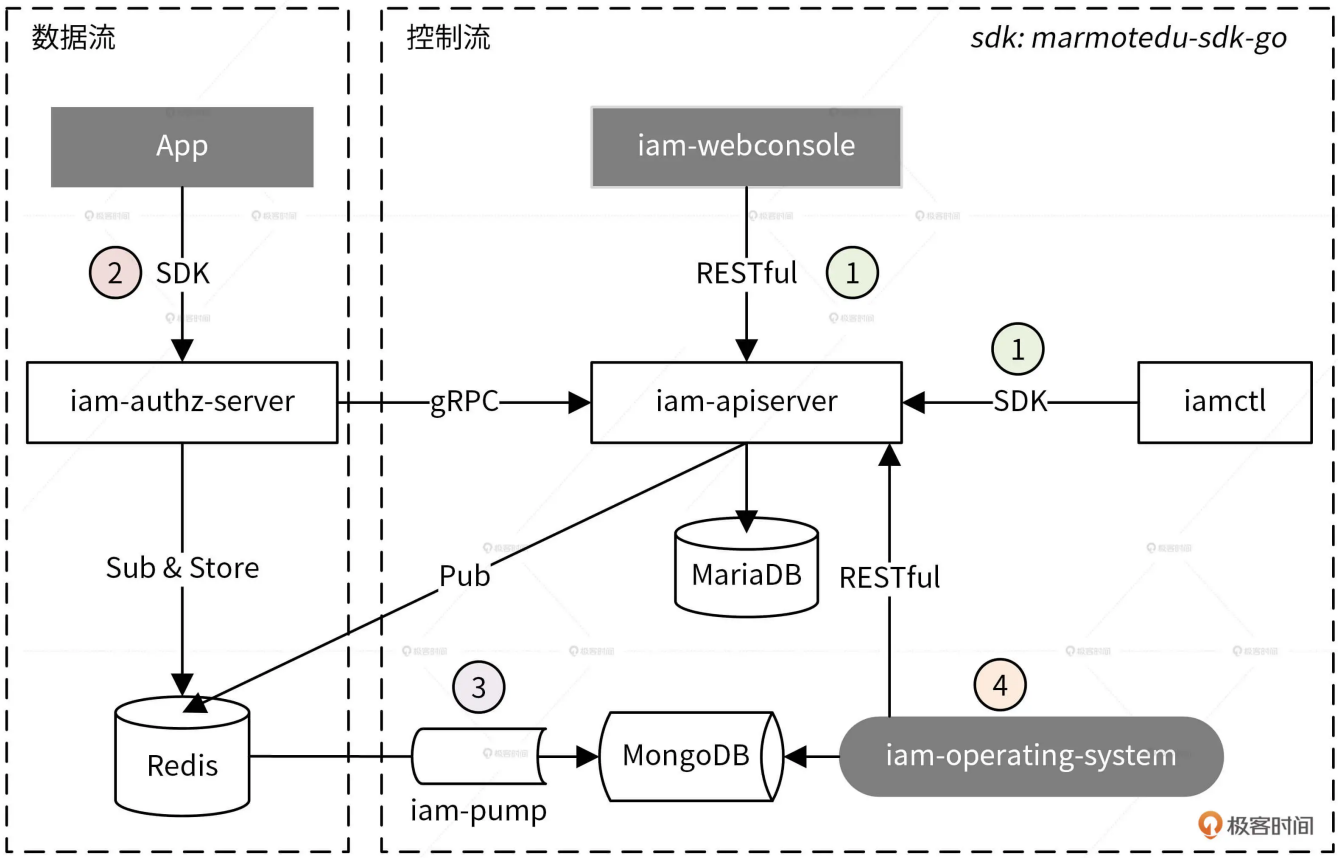
用户（User）：实现对用户的增、删、改、查、修改密码、批量修改等操作。

密钥（Secret）：实现对密钥的增、删、改、查操作。

策略（Policy）：实现对策略的增、删、改、查、批量删除操作。

## IAM 系统的架构长啥样？

知道了 IAM 的功能之后，我们再来详细说说 IAM 系统的架构，架构图如下：




IAM 系统的完整架构

总的来说，IAM 架构中包括 9 大组件和 3 大数据库。我将这些组件和功能都总结在下面的表格中。这里面，我们主要记住 5 个核心组件，包括 iam-apiserver、iam-authz-server、iam-pump、marmotedu-sdk-go 和 iamctl 的功能，还有 3 个数据库 Redis、MySQL 和 MongoDB 的功能。

| 序号 | 分类 | 组件 | 功能说明 |
| --- | --- | --- | --- |
| 1 | 核心组件 | iam-apiserver | 核心组件，通过 RESTful API 完成用户、密钥和授权策略的增删改查 |
| 2 | | iam-authz-server | 授权服务，从 iam-apiserver 拉取密钥和授权策略，并缓存在内存中，用户通过请求 iam-authz-server 提供的 /v1/authz 接口来完成资源的授权。/v1/authz 接口会查询缓存的授权策略，根据这些策略决定授权是否通过。iam-authz-server 也会将授权日志上报的 Redis 中 |
| 3 | | iam-pump | 从 Redis 中拉取缓存的授权日志，分析后存入 Mongo 数据库中 |
| 4 | | marmotedu-sdk-go | IAM 系统的 golang sdk， 参考了 Kubernetes 的 client-go，封装了 iam-apiserver 和 iam-authz-server 的所有 RESTful API接口，方便用户调用 |
| 5 | | iamctl | IAM 系统的客户端，参考了 Kubernetes 客户端工具 kubectl的实现，通过 marmotedu-sdk-go 访问 iam-apiserver 和 iam-authz-server。iamctl 封装了 iam-apiserver 的所有 RESTful API，还封装了其它功能。用户可以通过命令行的方式访问 iam-apiserver |
| 6 | 旁路组件 | app | 第三方应用，是 IAM 系统的使用方，通过 RESTful API 或者 marmotedu-sdk-go 调用 iam-authz-server 提供的 /v1/authz 接口完成资源的授权 |
| 7 | | iam-webconsole | IAM 系统的前端，通过 RESTful API 调用 iam-apiserver 实现用户、密钥和策略的增删改查 |
| 8 | | iam-operating-system | IAM 运营系统，可以用来展示运营数据或者对 IAM 进行运营类管理，比如提供上帝视角查看所有用户的资源，调整某个用户下密钥的最大个数等 |
| 9 | | Loadbalance | 负载均衡器，可以是 Nginx、Haproxy 或者 API 网关，后端挂载多个 iam-apiserver 和 iam-authz-server 实例，实现 iam-apiserver 和 iam-authz-server 组件的高可用 |

前 5 个组件是我们需要实现的核心组件。
后 4 个组件是一些旁路组件，不影响项目的使用。
如果感兴趣，你可以自行实现

此外，IAM 系统为存储数据使用到的 3 种数据库的说明如下所示。

| 数据库 | 功能说明 |
| --- | --- |
| Redis | 缓存数据库，用来缓存密钥和授权策略，降低访问延时，<br>同时也会缓存授权日志作为运营系统的数据来源 |
| MySQL | 持久性存储用户、密钥和授权策略信息 |
| MongoDB | 存储授权日志，供后期运营系统展示和分析 |

## 通过使用流程理解架构

只看到这样的系统架构图和核心功能讲解，你可能还不清楚整个系统是如何协作，来最终完成资源授权的。所以接下来，我们通过详细讲解 IAM 系统的使用流程及其实现细节，来进一步加深你对 IAM 架构的理解。总的来说，我们可以通过 4 步去使用 IAM 系统的核心功能。

### 第 1 步，创建平台资源。

用户通过 iam-webconsole（RESTful API）或 iamctl（sdk marmotedu-sdk-go）客户端请求 iam-apiserver 提供的 RESTful API 接口完成用户、密钥、授权策略的增删改查，iam-apiserver 会将这些资源数据持久化存储在 MySQL 数据库中。而且，为了确保通信安全，客服端访问服务端都是通过 HTTPS 协议来访问的。

### 第 2 步，请求 API 完成资源授权。

用户可以通过请求 iam-authz-server 提供的 /v1/authz 接口进行资源授权，请求 /v1/authz 接口需要通过密钥认证，认证通过后 /v1/authz 接口会查询授权策略，从而决定资源请求是否被允许。

为了提高 /v1/authz 接口的性能，iam-authz-server 将密钥和策略信息缓存在内存中，以便实现快速查询。那密钥和策略信息是如何实现缓存的呢？

首先，iam-authz-server 通过调用 iam-apiserver 提供的 gRPC 接口，将密钥和授权策略信息缓存到内存中。同时，为了使内存中的缓存信息和 iam-apiserver 中的信息保持一致，当 iam-apiserver 中有密钥或策略被更新时，iam-apiserver 会往特定的 Redis Channel（iam-authz-server 也会订阅该 Channel）中发送 PolicyChanged 和 SecretChanged 消息。这样一来，当 iam-authz-server 监听到有新消息时就会获取并解析消息，根据消息内容判断是否需要重新调用 gRPC 接来获取密钥和授权策略信息，再更新到内存中。

**第 3 步，授权日志数据分析。**

iam-authz-server 会将授权日志上报到 Redis 高速缓存中，然后 iam-pump 组件会异步消费这些授权日志，再把清理后的数据保存在 MongoDB 中，供运营系统 iam-operating-system 查询。

这里还有一点你要注意：iam-authz-server 将授权日志保存在 Redis 高性能 key-value 数据库中，可以最大化减少写入延时。不保存在内存中是因为授权日志量我们没法预测，当授权日志量很大时，很可能会将内存耗尽，造成服务中断。

**第 4 步，运营平台授权数据展示。**

iam-operating-system 是 IAM 的运营系统，它可以通过查询 MongoDB 获取并展示运营数据，比如某个用户的授权 / 失败次数、授权失败时的授权信息等。此外，我们也可以通过 iam-operating-system 调用 iam-apiserver 服务来做些运营管理工作。比如，以上帝视角查看某个用户的授权策略供排障使用，或者调整用户可创建密钥的最大个数，再或者通过白名单的方式，让某个用户不受密钥个数限制的影响等等。

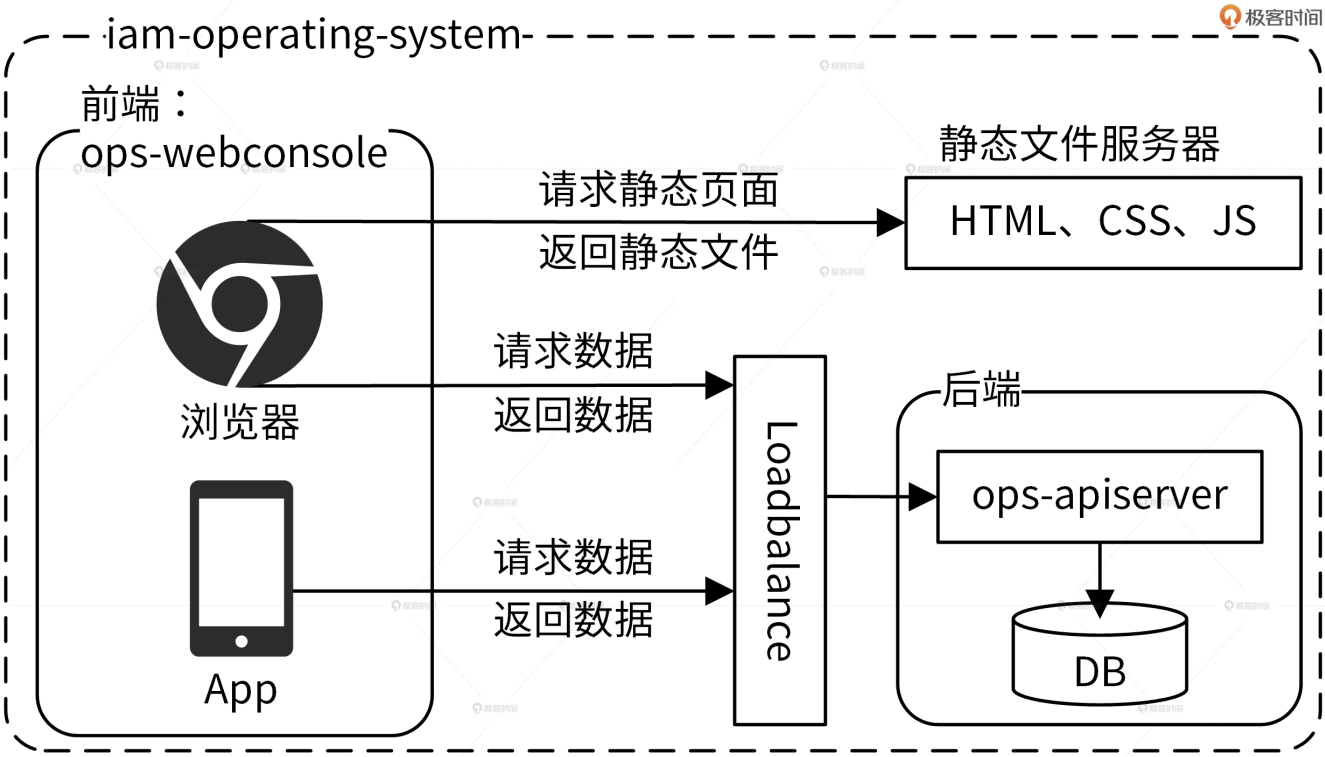
## IAM 软件架构模式

在设计软件时，我们首先要做的就是选择一种软件架构模式，它对软件后续的开发方式、软件维护成本都有比较大的影响。因此，这里我也会和你简单聊聊 2 种最常用的软件架构模式，分别是前后端分离架构和 MVC 架构。

### 前后端分离架构

因为 IAM 系统采用的就是前后端分离的架构，所以我们就以 IAM 的运营系统 iam-operating-system 为例来详细说说这个架构。一般来说，运营系统的功能可多可少，对于一些具有复杂功能的运营系统，我们可以采用前后端分离的架构。其中，**前端负责页面的展示以及数据的加载和渲染，后端只负责返回前端需要的数据。**

iam-operating-system 前后端分离架构如下图所示。




采用了前后端分离架构之后，当你通过浏览器请求前端 ops-webconsole 时，ops-webconsole 会先请求静态文件服务器加载静态文件，比如 HTML、CSS 和 JavaScript，然后它会执行 JavaScript，通过负载均衡请求后端数据，最后把后端返回的数据渲染到前端页面中。

采用前后端分离的架构，让前后端通过 RESTful API 通信，会带来以下 5 点好处：

可以让前、后端人员各自专注在自己业务的功能开发上，让专业的人做专业的事，来提高代码质量和开发效率

前后端可并行开发和发布，这也能提高开发和发布效率，加快产品迭代速度

前后端组件、代码分开，职责分明，可以增加代码的维护性和可读性，减少代码改动引起的 Bug 概率，同时也能快速定位 Bug

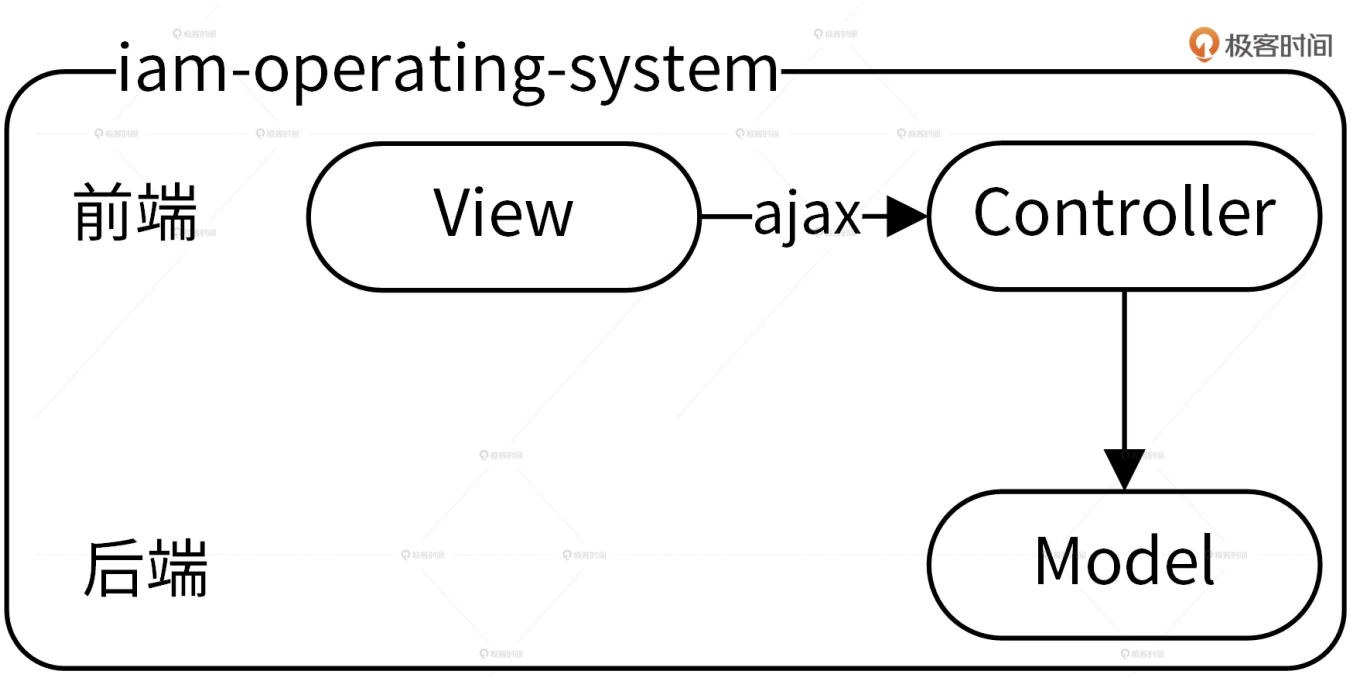
前端 JavaScript 可以处理后台的数据，减少对后台服务器的压力

可根据需要选择性水平扩容前端或者后端来节约成本

### MVC 架构

但是，如果运营系统功能比较少，采用前后端分离框架的弊反而大于利，比如前后端分离要同时维护 2 个组件会导致部署更复杂，并且前后端分离将人员也分开了，这会增加一定程度的沟通成本。同时，因为代码中也需要实现前后端交互的逻辑，所以会引入一定的开发量。

这个时候，我们可以尝试直接采用 MVC 软件架构，MVC 架构如下图所示。




MVC 的全名是 Model View Controller，它是一种架构模式，分为 Model、View、Controller 三层，每一层的功能如下：

View（视图）：提供给用户的操作界面，用来处理数据的显示。

Controller（控制器）：根据用户从 View 层输入的指令，选取 Model 层中的数据，然后对其进行相应的操作，产生最终结果。

Model（模型）：应用程序中用于处理数据逻辑的部分。

**MVC 架构的好处是通过控制器层将视图层和模型层分离之后，当更改视图层代码后时，我们就不需要重新编译控制器层和模型层的代码了。**同样，如果业务流程发生改变也只需要

变更模型层的代码就可以。在实际开发中为了更好的 UI 效果，视图层需要经常变更，但是通过 MVC 架构，在变更视图层时，我们根本不需要对业务逻辑层的代码做任何变化，这不仅减少了风险还能提高代码变更和发布的效率。

除此之外，还有一种跟 MVC 比较相似的软件开发架构叫三层架构，它包括 UI 层、BLL 层和 DAL 层。其中，UI 层表示用户界面，BLL 层表示业务逻辑，DAL 层表示数据访问。在实际开发中很多人将 MVC 当成三层架构在用，比如说，很多人喜欢把软件的业务逻辑放在 Controller 层里，将数据库访问操作的代码放在 Model 层里，软件最终的代码放在 View 层里，就这样硬生生将 MVC 架构变成了伪三层架构。

这种代码不仅不伦不类，同时也失去了三层架构和 MVC 架构的核心优势，也就是：**通过 Controller 层将 Model 层和 View 层解耦，从而使代码更容易维护和扩展**。因此在实际开发中，我们也要注意遵循 MVC 架构的开发规范，发挥 MVC 的核心价值。

## 总结

一个好的 Go 应用必须要保证应用的安全性，这可以通过认证和授权来保障。也因此认证和授权是开发一个 Go 项目必须要实现的功能。为了帮助你实现这 2 个功能，并借此机会学习 Go 项目开发，我将这 2 个功能升级为一个 IAM 系统。通过讲解如何开发 IAM 系统，来教你如何开发 Go 项目。

因为后面的学习都是围绕 IAM 系统展开的，所以这一讲我们要重点掌握 IAM 的功能、架构和使用流程，我们可以通过 4 步使用流程来了解。

首先，用户通过调用 iam-apiserver 提供的 RESTful API 接口完成注册和登录系统，再调用接口创建密钥和授权策略。

创建完密钥对和授权策略之后，IAM 可以通过调用 iam-authz-server 的授权接口完成资源的授权。具体来说，iam-authz-server 通过 gRPC 接口获取 iam-apiserver 中存储的密钥和授权策略信息，通过 JWT 完成认证之后，再通过 ory/ladon 包完成资源的授权。

接着，iam-pump 组件异步消费 Redis 中的数据，并持久化存储在 MongoDB 中，供 iam-operating-system 运营平台展示。

最后，IAM 相关的产品、研发人员可以通过 IAM 的运营系统 iam-operating-system 来查看 IAM 系统的使用情况，进行运营分析。例如某个用户的授权 / 失败次数、授权失败时的授权信息等。

另外，为了提高开发和访问效率，IAM 分别提供了 marmotedu-sdk-go SDK 和 iamctl 命令行工具，二者通过 HTTPS 协议访问 IAM 提供的 RESTful 接口。

## 课后练习

1. 你在做 Go 项目开发时经常用到哪些技能点？有些技能点是 IAM 没有包含的？
2. 在你所接触的项目中，哪些是前后端分离架构，哪些是 MVC 架构呢？你觉得项目采用的架构是否合理呢？

期待在留言区看到你的思考和分享，我们下一讲见！

分享给需要的人，Ta订阅后你可得 24 元现金奖励

赞 11 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 开篇词 | 从 0 开始搭建一个企业级 Go 应用

下一篇 02 | 环境准备：如何安装和配置一个基本的 Go 开发环境？

## 精选留言 (39)

写留言

**素衣绾绾**
2021-05-27

就是说，项目的源码已经完成，然后作者会在专栏中带着我们一起搭建一个web项目框架的雏形，也就是专栏中说的一些目录规范，api设计规范，错误包设计等，具体的功能代码，就是照着已经写好的代码挑选其中核心的功能进行讲解，是吗？

> 作者回复: 专栏会介绍如何开发Go项目，demo项目只是专栏介绍的开发技术的一个落地项目。
>
> 二者是包含与被包含的关系，专栏包含了实战项目，Go开发方法和思路，一些知识点的讲解，以及我的一些研发经验和建议。
>
> 只看源码，你看到的仅仅是一个源码，但是很可能不了解源码背后的构建思路以及一些其它需要注意的地方，比如说只看源码，你仍然不知道开发规范，仍然不知道用了哪些设计模式和技巧。
>
> 所以建议以专栏学习为主，代码阅读为辅。通过代码去验证专栏的设计思路。

2 19

**tiny**

2021-05-26

沙发，跟作者也算半个同事

展开

9

**Paualf**

2021-06-02

在做项目的时候，项目间通信经常使用的是kafka这类消息队列，后台会经常用搜索功能，这部分数据都会放在了ES中如搜索，日志或者后台系统的一些搜索。IAM系统中使用了Redis做消息队列，使用了MongoDB去进行存储和查询，这些是我目前想到不一样的技能点。

...

展开

1 8

**、荒唐_戏_**

2021-06-03

完蛋了，上来一个iam系统就没理解😟

展开

作者回复: iam系统主要完成认证和授权功能。
认证：用来判断是否是平台的合法用户，例如用户名和密码就是认证的一种方式
授权：用来判断，是否可以访问平台的某类资源。

认证和授权 2个功能可以抽象成一个系统，这个系统名我起名为iam。类似于aws的iam，腾讯云的cam和阿里云的ram。

建议你参考下腾讯云的cam以协助你理解：https://cloud.tencent.com/document/product/599/40011。

不难，花几分钟相信你就会理解了。

5

**dll**

2021-06-04

不太明白，怎么同意这个用户创建并给密钥对，并给到对应的权限策略，这个步骤不需要人来同意吗？如果谁都能注册自己选择权限，那不是很危险？这个aim和jwt有什么关系

展开

作者回复: 管理员注册iam系统，并登录系统，创建密钥对和策略。密钥对分发给其他人，其他人通过密钥对进行资源授权，授权时填写自己的用户名就可以了。

Jwt是iam认证的方式

3

**helloworld**

2021-06-09

iam这个认证与资源授权系统和网关的认证与鉴权功能有何本质区别呢

作者回复: 跟网关的认证功能是一样的，iam的认证功能也是参考了tyk网关的实现。

跟网关的鉴权还是有些区别，iam是对资源鉴权，网关是对api鉴权，就是谁能或不能调用哪个api接口。

2

**老妖纣王**

2021-06-02

老师，您好，正好讲到iam系统有几个问题想请教你一下，在项目中使用认证授权，遇到几个疑惑的问题一直困扰着我？
假如APP是内部应用的场景，流程是用户通过登录API直接获取token，然后每次携带token进行请求，网关拦截进行token的认证。
（1）这里面token是保存在客户端的，是否安全？...

展开

作者回复: 1. 使用客户端用用户名登录后，证明客户端短时间是安全的，token默认有2小时的过期时间。

2. 这个刷新成本服务端还是能抗住的。如果token过期就只能重新登录了，至于多久过期，每个业务会根据需要自己选择，但很多开源软件一般默认2小时过期。

3. 密钥保存在客户端，需要客户端使用者去保证密钥不泄露，服务端也会保存一份密钥，用来对请求进行再加密，并和客户端传来的token对比，如果一致则请求通过。

2

**navy**
2021-05-27

令飞老师，腾讯云的CAM和IAM系统有什么区别吗？

展开

作者回复: 授权模型不一样。
CAM是ABAC。
IAM授权模型类似于RBAC和ACL

2

**nio**
2021-05-27

IAM只是我们学习这个专栏的小工具，主要还是借助这个项目学习go语言项目的设计及思想以及具体实现，其中这些理论能力会体现在项目的各个实现细节中，是这个意思吧

展开

作者回复: Nio理解的很正确!

2

**lei.guo**
2021-07-06

authz，这个名字有啥含义么？主要是这个z

展开

作者回复: 认证：authentication，缩写：authn
授权：authorization，缩写：authz

1

**helloworld**
2021-06-24

是不是可以理解为每一个使用iam系统作为认证和资源授权的业务应用，该业务应用的用户每次访问该业务应用的接口都要先经过iam系统，认证和资源鉴权通过后，才能继续访问该业务应用接口呢，也就是iam是业务应用的网关代理？

作者回复: 理解的没毛病，可以理解为一个授权webhook。

1

**helloworld**
2021-06-24

IAM能和网关结合使用吗，还是必须二选一呢

展开

作者回复: 网关做的是接口鉴权，也就是哪个用户/密钥可以访问哪个接口。而IAM做的是资源鉴权。例如淘宝中的iphone资源等

1

**helloworld**
2021-06-10

iam可以用于微服务吗，就是n多个应用使用同一套iam为各自应用内的资源对用户授权

作者回复: 可以的哦，它就是个微服务

1

**Vackine**
2021-05-28

iam 存储的密钥是创建了之后就是静态的一直存在mysql的么？会不会有密钥截获的风险？还是每个密钥有超时时间？

作者回复: 一直保存在mysql数据库中，登录用户名密码登录系统获取临时密钥保存在客户端，默认2小时过期。后面的通信全都走https，通过认证，https，过期时间来保证安全性

1

**冯甜莉**
2021-05-27

api-server提供全部的资源接口，基于api-server 公开RestFulApi完成授权存储sql，独立auth服务与api-server通过GRPC同步授权策略，完成资源授权；

肯定存在授权策略的构成多元需求带来的复杂性，资源授权占用内存及同步失败补偿措施，多台服务同步交互机制。

1

**小屎丸**
2021-07-11

为啥选择 MongoDB 作为日志数据分析展示库，怎么考虑的？谢谢

作者回复: 1. mongodb带有数据聚合功能，在某些场景下可以实现复杂的数据统计
2. 而且字段增减随意，查询方便
3. IAM系统授权日志量不大，场景也不复杂，再加上，这里是想展示mongodb的教学，所以就采用了mongo

1

**awmthink**
2021-07-05

老师，初步接触Web业务开发，对这一节中 IAM 系统的资源授权的流程图有2个疑问：
1) 步骤2中的创建密钥对，我理解其实就是JWT吧，只是一个token，和用户名一起构成了密钥对吗？
2) 图片步骤4，用密钥对去访问授权接口，这个不是用户直接访问吧，应该是用户用密钥对去访问资源时，由资源实时控制服务去调用authzServer来验证用户是否有权限来访问...
展开

作者回复: 1) 密钥对还不是JWT，密钥对包含secretId，secretKey。

secretKey可以用来签发Token（JWT），secretId可以使服务端根据secretId查找服务端保存的secretKey，反签发Token，对比2个token是否一致，完成认证。

2）用密钥对访问授权接口，这里的密钥对可以签发JWT Token，iam-authz-server判断该Token是否能访问/v1/authz接口。如果能访问，接下来才会进行资源授权的流程。一个是认证，一个是资源授权。

**Fis.**
2021-07-02

我理解authz是面向用户的借口，apiserver是核心处理模块，sdk不直接对接apiserver吗？authz和api server分离设计的优点是什么？另外authz从redis获取数据，那不是有可

能读到过期数据？（比如当时写入请求还在写入mysql过程中，还没同步到redis）

作者回复: sdk可以对接iam-apiserver和iam-authz-server。
authz和api server分离的优点，其实是：数据流和控制流的分离。

这样做的好处如下：

1. 控制流主要是用来做资源的CURD，如果控制流服务出故障，会影响用户的使用，但不会影响用户的业务。
2. 数据流出故障会影响用户的业务。

对于一个系统来说，影响用户业务是很严重的事情，而且我们日常变更最多的是控制流服务，控制流和数据流分离，使得我们发布变更控制流服务时，不会影响到数据流服务，也即不会影响用户的业务。

1

**乳酸菌**
2021-06-28

始终没有理解MVC 和三层架构的区别，老师有例子能加深理解吗？

作者回复: 重要区别：
MVC总的controller层不做具体的业务逻辑处理，主要用来做请求逻辑分发，可以把业务逻辑具体实现放在类似service这样的包中。在controller引用这个包。
三层架构中的BLL层，是用来做业务处理的。

还有其它区别，具体可以google下。

**Geek_a4cca6**
2021-06-28

我们现在使用的是离线工具生成yaml文件，符合openapi3.0规范，开发有些也用makdown，现要求全部改掉